(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
30 January 2003 (30.01.2003)

PCT

(10) International Publication Number
WO 03/009071 A1

(51) International Patent Classification⁷: G05B 19/418

(21) International Application Number: PCT/US01/22833

(22) International Filing Date: 20 July 2001 (20.07.2001)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: APPLIED MATERIALS, INC. [US/US]; P.O. Box 450A, Santa Clara, CA 95052 (US).

(72) Inventor: YUAN, Huey-Shin; 10319 McKlintock Lane, Cupertino, CA 95014 (US).

(74) Agents: BERNADICOU, Michael, A. et al.; Blakely, Sokoloff, Taylor & Zafman LLP, 7th floor, 12400 Wilshire Boulevard, Los Angeles, CA 90025 (US).

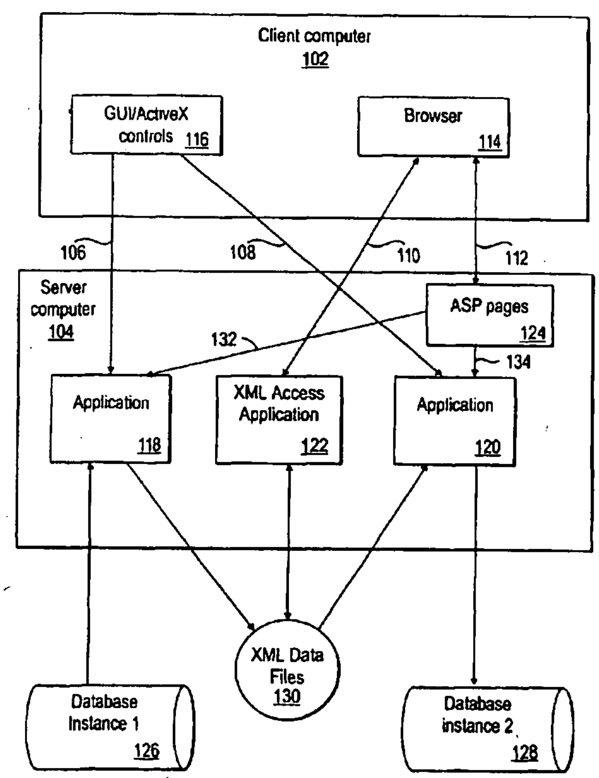
(81) Designated States (*national*): CN, JP, KR, SG.

(84) Designated States (*regional*): European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR).

Published:
— *with international search report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: SYSTEM AND METHOD FOR EXPORTING OR IMPORTING OBJECT DATA IN A MANUFACTURING EXECUTION SYSTEM



(57) Abstract: XML is used to express model object data for an MES application model object. The data for the model object is held in a database. To export information, the database definition is used to construct XML tag information and the database values are used to construct XML element contents. To import information, the XML tag information is used to construct database references, and the XML element contents are used to load the database. To export information, database address information names are used to form XML tag labels, and the model object data in the database is used to form XML element contents. New classes are created that are derived from the model object class. The new classes inherit the functionality of the model object classes but they also include new import and export methods that effectively wrap, or layer, the import or export logic around the model objects. DTDs may be used to validate XML documents that are exported or imported from such a system. Code generation facilitates the creation of the new class definitions and the DTD. The code generation considers the database metadata in generating the code.

WO 03/009071 A1

System and Method of Exporting or Importing Object Data In a Manufacturing Execution System

Background of the Invention

1. Field of the Invention

This invention relates to Manufacturing Execution Systems (MES) and, more particularly, to importing and/or exporting MES model object data.

2. Discussion of Related Art

Modern manufacturing facilities that manufacture semiconductor and other complicated components use Manufacturing Execution Systems (MES) to record and coordinate manufacturing events in real time. Among other things, these systems integrate and manage information generated by various activities and automated systems. MES functionality includes (1) batch/lot management, (2) real-time operations management, (3) quality management, (4) resource management (including scheduling, dispatching, equipment monitoring, and preventive maintenance), (5) direct and indirect labor coordination and dispatching, (6) specifications and work instructions, (7) recipe management, (8) detailed material control and genealogy, (9) actual cost tracking, and the like.

An MES, at some level of abstraction, models the manufacturing operations that it helps monitor and coordinate. An MES is typically an object-oriented application that often exposes the behavior of the application, and specifically the application's objects, through an interface. The modeling effort is costly and time consuming but highly important to the success of the MES. Because these manufacturing operations and related activities tend to be complex, many modern Manufacturing Execution Systems employ databases to hold various types of object data. Moreover, standards have been developed to describe, at some level of abstraction, the interfaces to the MES components. For example, CIM is a standard known in the MES art for describing a model of the structural setup of a manufacturing operation, e.g., machines, materials, how to process materials, and the like.

Many modern manufacturing companies first create a "development" system when considering a new manufacturing line or portion thereof. These development systems often include similar physical components and systems to the eventual, desired production system, including for example the same or similar MES. The development system is used to construct, test and tune a model in the MES.

Often, however, the production system (or target system) differs from the development system in some aspects. For example, the production system may use a certain physical component that has a corresponding physical component in the development system but the development component and the production component differ in some attributes, e.g., execution parameters. Sometimes this difference is inevitable because the production components are too scarce or expensive to use in a development system. Alternatively, the production component may not yet exist in a "released" form, yet manufacturers nonetheless benefit from creating and modeling a production system in anticipation of its eventual release. Consequently, though a development system is often similar to an eventual production system, invariably some differences exist.

Typically, migration of the data from a development system to a production system is *ad hoc* and relies on the use of a skilled technician. The technician must first understand an interface to the MES of interest, e.g.., the one from which data is to be exported from, and then he or she must write some form of script to invoke the model objects to obtain the data. This approach is time consuming, costly, and error prone. Moreover, if the model changes, the script may no longer work (e.g., it may refer to now non-existent or changed objects).

Summary

One aspect of the invention uses XML to express model object data for an MES application model object. The data for the model object is held in a database. To export information, the database definition is used to construct XML tag information and the database values are used to construct XML element contents. To import information, the XML tag information is used to construct database references, and the XML element contents are used to load the database.

Under another aspect of the invention, logic parses an XML expression and converts XML tag labels to database address information, and converts XML element contents to database values.

Under yet another aspect of the invention, a class is derived from an MES model object class, and the class so derived inherits all functionality from the MES model object class. The class further includes import method logic to receive an XML expression of model object data and load the XML-expression of the data in the database.

Under another aspect of the invention, a class that is derived from an MES model object class, and the class so derived inherits all functionality from the MES model object class. The class further includes export method logic to create an XML expression of model object data in the database.

Under another aspect of the invention, data may be imported into an MES application having at least one model object, by forming an XML expression of the object data. The the XML expression is sent to the MES application, and the MES application parses the XML expression to determine the object type, the object parts, and the object data values. The MES application loads the determined object type, the object parts, and the object data values into a database corresponding to the MES application.

Under another aspect, a DTD expression of the object class definition is formed and the
XML expression is validated against the DTD expression.

Under another aspect of the invention, import logic is layered over the model object, and the import logic defines an external interface for loading model object data. The import logic includes logic to invoke the model object to load the model object data into the database.

Under another aspect of the invention, export logic is layered over the model object, the export logic defines an external interface for obtaining model object data.

The export logic includes logic to invoke the model object to obtain the model object data from the database.

Under another aspect of the invention, a MES includes a client and server computer. The client has a user interface control for exporting or importing selected model objects of the MES and for constructing a client request to request export or import of selected objects. The server has the MES application executing thereon. The MES application is organized as a relationship of model objects, with each model object having data therefor stored in a database. The application includes import and export logic, responsive to the client request, for forming an XML expression of model object data in response to an export request, and for loading data expressed in XML form into the database in response to an import request.

Brief Description of the Drawing

In the Drawing,

Figure 1 is an architectural diagram of a preferred embodiment of the invention;

Figure 2 is a flow chart showing the logic for exporting information according to a preferred embodiment of the invention;

Figure 3 is flow chart showing the logic for reading information from a database and writing an XML-based version thereof according to a preferred embodiment of the invention;

Figure 4 is a flow chart showing the logic for importing information according to a preferred embodiment of the invention;

Figure 5 is flow chart showing the logic for reading information from an XML document and writing corresponding information to a database according to a preferred embodiment of the invention;

Figures 6A-6B show class and object expressions for layered export and import software of a preferred embodiment; and

Figure 7 is a system diagram of exemplary hardware for a client or server computer according to exemplary embodiments of the invention.

Detailed Description

Overview

Preferred embodiments of the invention facilitate the exportation and importation of MES objects modeling a manufacturing environment. A model object may contain many parts, such as product, material, lot templates linked to process flows, and the like. Likewise, the model information may define the data interrelationships. In one embodiment, though the MES model is object-oriented, the data for the various objects is held in a table-based database, such as ones available from Oracle Corporation.

A preferred embodiment of the invention facilitates importation and exportation of model data by layering importation and exportation logic over the model. In this fashion, the importation and exportation logic works, even if the underlying model is changed. That is, no tools need to be redeveloped or modified because the model changes, and system administrators do not need to track model changes to ensure that importation and exportation functionality works.

Other preferred embodiments use a human readable representation of the model object data to facilitate exportation and/or importation of model data. For example, extensible markup language (XML) may be used as described below to describe the model object data as an XML document. A metadata-like description of database information and of the model object class definitions may be defined as a Document Type Definition (DTD). (Both XML and DTDs are known and described by w3c and other organizations.) The use of XML and DTDs facilitates exportations and importation of model object data in several ways. For example, because meaningful markup tags are used, the data in the XML document is self-describing and easier to comprehend by humans and/or tools. Moreover, the use of DTD facilitates the use of an XML document by defining valid structure so that XML documents may be parsed to test their structural validity. Moreover, XML and DTDs are based on well defined

standards, and thus tools may be more easily developed to modify or create information to be imported and/or exported from a system.

In short, under the principles of a preferred embodiment, a user identifies MES model object(s) of a source environment to be exported (e.g., specific instances or all such objects) and, through a graphical user interface (GUI), may requests that it (they) be exported. Software, layered over the model objects of the source environment, responds by reading and extracting the relevant information from a database holding the object data and writing the information into an XML document having a DTD corresponding to the selected object class definition. The XML document may then be accessed and/or modified with other tools. For example, an XML editor may be used to view exported information and to allow a user to modify that data as necessary so that it may be imported into another system. Alternatively, other tools may construct XML-forms of information that should be imported, or loaded, into a target system by referring to a DTD that defines the structure of data that may be imported into a system. In any event, an XML document may be used to hold object data to be imported into a target environment. Other software, layered over the model objects of the target environment, read the XML information, effectively deconstruct it into its parts, and load the information in the target database. The DTD may be used to provide one measure of information validation (i.e., DTD validation against the XML document, as known in the art) before subsequent database validation by the target system.

Architecture

Figure 1 shows an exemplary system on which preferred embodiments of the invention may be manifested. A client computer 102 may communicate with a server computer over a communication network, represented collectively by the paths 106, 108, 110, and 112.

The client 102 includes a browser 114. In one embodiment, the browser is an Explorer browser, available through Microsoft Corporation. This one embodiment uses an ActiveX control (implemented with conventional technology available through

Microsoft Corporation) to provide certain GUI functionality described below. Though no arrow is shown between the Active X control 116 and the browser 114, these two components cooperate in a known way. Other browser and GUI control techniques may be substituted under the principles of the invention without loss of generality.

The server 104, as shown, includes a first application 118, a second application 120, and XML access application 122, and at least one active server page 124. The first application 118 operates with a first database instance 126, which for purposes of this description can be considered as a source database instance, representative of a development environment. The second application 120 operates with a second database instance 128, which for purposes of this description can be considered as a target database instance, representative of a production environment. The first and second applications 118, 120 and the XML access application 122 all may communicate with at least one XML document 130, which may be manifested as a file, though it also may be manifested in other forms. The active server page124 provides one way (not the only way) to access the server applications 118, 120 in response to commands from the browser 114. Figure 1 among other things illustrates that information may be exported from one type of MES application 118 and imported into another 120 on the same server, but it should be appreciated that the various applications shown on the one server may, and indeed are expected to be, distributed among several servers. For example, each MES application may have its own server.

The first and second applications are object-oriented MES applications constructed according to conventional techniques. The actual type of MES applications and their MES functions are not material to the invention and thus their detailed logic is not described. In one embodiment, the MES objects (also referred to herein as model objects, or domain objects) are constructed as COM + components with published APIs. (COM + is a known approach of expressing object behavior and interfaces.) The MES object data is held in persistent form in the corresponding database (e.g., application 118 has data in database instance 126).

In one embodiment, the database instances 126, 128 organize model object data as tables. For example, visualizing a simple table, the leftmost column may correspond

to a class, and each column may correspond to a member of the class definition. Each row in the exemplary table would then correspond to an object instance, in which the actual values for a given row and column correspond to that instance's member's value. Naturally, the data relationships may define further levels of hierarchy and references as is known in the art. The operative issue is that the metadata of the table definitions corresponds to MES model class definitions and that mechanisms are known for getting and setting object information held in the object instance and/or the corresponding database entities. As is known in the art, the metadata for the database may express that a given column holds integer values, character data, a reference to another database entity, etc.

The XML access application 122 may be an XML editor or other tool that can create and/or present XML documents as described below. For example, an XML editor may be used to modify an XML document that was exported from one application, or another tool may be used to create an XML document consistently with a DTD.

The active server page124, in an exemplary embodiment, is a Microsoft Active Server Page. The ASP 124 provides one way for accessing the applications by paths 132, 134, whereas paths 106 and 108 provide another (in which information is passed to the applications as COM+ arguments).

Logic

Figure 2 is a flowchart showing the logic for exporting information according to a preferred embodiment of the invention. The logic starts 200, and with the assistance of the browser (114, Fig. 1) a user selects objects to be exported 205. In one embodiment, the ActiveX control (116, Fig. 1), with the assistance of auto-discovery techniques, gathers a list of the model object types (i.e., classes) that support export and/or import functionality. This list is then presented to a user to identify the types of objects that may be exported. The user may then select all objects of a given object type or just specific identified instances. The user also provides a filename in which an XML document representation of the model object data should be held. In response to the

selection, the Active X control (116, Fig. 1) locates the appropriate server application, in this example the first application (118, Fig. 1), and the client sends the export request 210 as COM+ API arguments through the Active X control. The server application (118, Fig. 1) validates that the request refers to valid database objects 215. The server application then accesses the relevant source database instance (126, Fig. 1) holding the information for the identified object, reads the information corresponding to the request, and writes that data 220 as an XML document (150, Fig.1). Once the exporting of information is completed, the first application (118, Fig. 1) returns 225 status information to the Active X control (116, Fig. 1). The status is then displayed 230 to the user via a browser (114, Fig. 1) to the user who initiated the exportation, and the logic ends 299.

Figure 3 is a flow chart more specifically showing the logic of reading objects from a database and writing it as an XML document as described above. The logic starts 300 and attempts to read 305 object data corresponding to the objects included in the client export request. As stated above, the client request may identify all or select objects of an identified object type, and an internal list representation of objects to be exported is constructed accordingly. The first application (118, Fig. 1) then determines 310 whether the identified object exists in the database to validate that portion of the request, i.e., the portion corresponding to the validated object. If the identified object does not exist, the application updates 315 an error information data structure and determines 320 whether the client request has identified other objects to be exported. If the identified object exists in the database (see 310), the first application (118, Fig. 1) reads 325 a complete version of the object's data from the database (126, Fig. 1). This reading is explained in further detail below. The collected information is then formatted and written 330 into an XML document (130, Fig. 1). More specifically, the XML document has tags to mark the data, in which the tag labels correspond to the database entity information and in which the contents correspond to the actual table data. Depending on the data organization, the process of reading may require iteration through the object data, and a check is made 335 to determine whether all the information of a given object being exported has been written to an XML document. As part of the exportation, other information is gathered and eventually reported, such as the number of records exported. If not all of the information has been gathered, the iteration

continues 330 to format the object data accordingly. If the object has been fully formatted and written to an XML document (130, Fig. 1), then a check is made 320 whether the exportation request has been completed, i.e., whether there are other objects to be exported. If not, as outlined above, a subsequent data object in the request is selected and the processing continues as described above. If the exportation request has been completed the processing ends 399.

The reading and formatting of the object data is greatly facilitated by layering the software over the model software. In particular, as outlined above the MES applications are composed of objects implementing the model provided by that application. These models include "get" and "set" functionality to get and set object state, following well known object-oriented programming convention. These gets and sets have the logic for accessing the underlying database as necessary and the get and set routines are published or otherwise known.

A preferred embodiment builds on the above as follows. A new export class is defined as a derived class of the actual model class. As such, the new export class inherits all of the functionality defined in the actual model class, including the gets and sets just mentioned. The new class defines a constructor method (constructor methods are known in the art) in which an object identifier is passed in as a variable and used to get and initialize object data members with the information corresponding to that object identifier. Thus, when an instance of this new class is instantiated it will be constructed with the data corresponding to the object identifier. An instance of the new export class may then in this fashion be constructed, passing in the identifier of the object to be exported, and an export object will be created that will be initialized to contain member data corresponding to the object to be exported. In effect, this portion of the new export object is a copy of the actual model object.

The new class also includes export method logic that creates the XML document in the following manner. Following the organization of the object's data in the database, an opening tag is written for the highest level database entity for that object, followed by the entities that that entity contains, which may be other entities or data. Closing tags and empty tags are inserted as appropriate, depending on the database contents. The

highest tag corresponds to the object class, and each other tag label corresponds to the label used in the database, i.e., the column name of a table. Appendix A shows an exemplary XML document created in this manner. In this example, "ENTITYDETAILS" corresponds to the object name or class, and the tags underneath correspond to the table column names. (This method is only called after other software writes prolog information to the document, identifying a corresponding DTD, a root element, and the like.)

The creation of the export logic is greatly facilitated through the use of code generation. A conventional code generator may be used to do the following. The code generator is passed the object name or class (e.g., ENTITYDETAILS in the above example) along with the table name corresponding to that object name, and along with a description of other entities that that object inherits from (in other words, other tables that may be referenced from one of the object members). The generators "script" uses the above to construct C++ code to do the following:

1. To call library primitives to write XML tags to a file (i.e., the one to hold the XML document) to start an XML element. The tag label will either be the passed in object name (for the highest level tag) or the column names of the table (which may be determined from the database's metadata).
2. To nest the logic of (1) as appropriate to handle hierarchical arrangements, see, e.g., Appendix A in which a tag "ENTITY" follows "ENTITY DETAILS".
3. To insert the appropriate calls to get, or obtain, the database values in the database table (i.e., this is done at the leaves of the structural relationship); referring to the example of Appendix A, after code is generated to make the calls to write the tag "<ENT_NAMESPACE>" code is generated to obtain the actual value at the specified table at the column "ENT_NAMESPACE" and to format the returned value accordingly to ASCII.
4. To write closing tags as appropriate.

The generation is facilitated in that the organization of the data is already known and expressed as database metadata.

The database metadata may be used to also generate a DTD expression of the metadata as well and thus generate a DTD for the object type at issue. See Appendix B

for an exemplary DTD corresponding to the object data expressed in XML document of Appendix A.

In some cases, the generated code (either export logic or DTD) may need developer modification because the database information is not easily translated into an XML form. For example, if the database information is a reference to another database entity, a developer may need to modify the automatically generated code to reflect and properly handle this type of entity.

Using the above, when a model is changed by a developer he or she may need to run a utility to generate the code as outlined above and to modify the generated code as appropriate. Regardless, the interface to the export logic remains the same, i.e., the export methods, and thus the tool to export object data does not need to change when the model changes. Instead, the export tool needs to only instantiate an object from the new export class (derived from the object class to be exported) and to pass to the constructor the object identifier for the object to be exported. The instantiated object will thus automatically have all of the logic to obtain any data of interest. Once constructed, that object's (i.e., the one from the derived class) export method is invoked to write the XML data. As outlined above, this will write the XML file. The corresponding DTD file for this object type was created at the time the export logic code was generated.

The exportation process at an object and class level is illustrated in Figure 6A, utilizing, in part, the known Booch notation.

Under the above embodiment, a DTD is created to represent the object data expressed in the XML document holding exported data. The DTD may also be used by other tools (e.g., XML editor) to facilitate the modification of such an XML document (e.g., for subsequent import in another embodiment) or to create an XML document automatically. (For example, a DTD may first be created based on the database metadata, and the DTD form may subsequently be used by code generators to create the XML document write commands to insert the tag information.) Moreover, though the example of Appendix B uses conventional XML syntax to show that the entities are parsed character data (PCDATA), it will be appreciated that more restrictive forms of

DTDs may be written, e.g., other data forms, such as integer, defining sequence relationships, and the like.

Figure 4 is a flowchart showing the logic for importing information according to the a preferred embodiment of the invention. The logic starts 400, and with the assistance of the browser (114, Fig. 1) a user selects objects to be imported 405 and specifies an importation policy, e.g., add information as specified in the request, or update information only if target database does not have the information. In response to the selection, the Active X control (116, Fig. 1) locates the appropriate server application, in this example the second application (120, Fig. 1) , and sends the import request 410 to that application. The importation request, among other things, identifies the XML document holding the data to be imported. This may be from the previously explained exportation process, an edited version thereof, or a document independently created. The server application (120, Fig. 1) validates 415 the XML document with reference to the DTD, i.e., that the XML document has the form defined in the DTD. The server application then accesses the XML document (130, Fig.1), processes the XML document data, and loads 420 the corresponding information into the identified target database instance (128, Fig. 1). Once the importing of information is completed, the second application (120, Fig. 1) returns 425 status information to the Active X control (116, Fig. 1). The status is then displayed 430 to the user via a browser (114, Fig. 1) to the user who initiated the importation, and the logic ends 499.

Figure 5 is a flow chart more specifically showing the logic of loading objects into a database based on the information in an XML document. The logic starts 500 by reading 505 information from the client importation request represented, in part, by the XML document (130, Fig.1) and collecting sufficient information to identify a first selected object in the request. The second application (120, Fig. 1) then attempts to read 510 the data corresponding to that object from the target database (128, Fig. 1). The software then determines 515 whether the identified object already exists in the target database. If it does exist, the application analyzes 520 the import policy specified in the client request to determine whether existing data should be over-ridden with the data from the XML document (130, Fig. 1). If the identified object exists in the database (see 515) and the client request indicates that the database information should not be

over-ridden (see 520) then the software iterates 525 through the XML document (130, Fig. 1) to determine if the document contains more data 525 for other objects to be imported. If either the identified object does not exist in the database (see 515) or if the client requested indicated that existing data should be over-ridden (see 520), the application reads 530 all of the information in the XML document (130, Fig. 1) corresponding to the identified object. The XML information collected for the document is validated 535 against the database. This validation in some cases will be more robust than that done at the DTD-level of validation. For example, with modern DTD, database metadata may be more restrictive than that expressible in DTD. Thus, the XML document may satisfy DTD validation but fail database validation. The application then determines 540 whether all of the data in the XML document is valid against the database metadata. If the data is not valid, the software accumulates 545 this error information in a data structure. If the data is valid, the software updates the target database (128, Fig. 1) with the object data compiled from the XML document. This update 550 includes updating all of the information related to the updated object, for example, in the case of hierarchical objects referring to other data and objects. Then, as outlined above, the software iterates 525 through the XML document (130, Fig. 1) to determine if the document contains more data. If so, the process repeat as described above; if not, the update process ends 599.

Similar to the export case described above, importation is facilitated by inheritance and code generation. Figure 6B illustrates the process. In analogous fashion to that described above, an import class is defined that is derived from the actual MES object class. The derived class, like the case above, includes generated and modified code, which in this case has calls to library primitives to parse the XML document to collect table references, e.g., table name and columns, and to collect the actual data from the tag contents. The generated code also has the logic to write to the database as needed. In some cases, the XML contents will need to be converted to a different type, e.g., integer, and in some cases this conversion may need to rely DTD definition and/or the metadata definition. Like the above, this code may need developer modification, but also like the above changes to the model will not require changes to the import tool.

In this case, the instantiated object of the new class is passed the identifier for the object that should receive the data so that that object may be initiated appropriately. That object may then be invoked with "gets" to see what state it has (e.g., see above in conjunction with import policy) and with "sets" as needed.

Figure 7 is a high-level diagram of a computer system that may be used for the client or servers shown in figure 1. The system 700 includes one or more processors 702, a memory 704 (potentially organized as virtual memory 706), display device 708, and input/out logic 710 in communication via a bus 712.

The above embodiments offer several advantages. By way of example only and not limitation, the embodiments allow XML documents and DTD information to be used to express object data, even in embodiments where the database is not object oriented. This allows the object data to be validated (in a DTD sense) before it is imported. In addition besides being more understandable because they are self-describing data, XML and DTD documents use standards that facilitate the development of tools. In addition, the import and export utility does not need to change when the model changes and instead continues to work. Exportation and importation are greatly benefited and can be largely automated. Users do not need to develop custom scripts for export or import, nor do they need to enter vast amounts of information manually. The tools may be implemented with known internet and intranet technologies. XML also facilitates the support of internationalization of the information, i.e., so that the intermediary form may be presented in multiple languages. In addition, since XML documents are not necessarily physical files, the XML document intermediary form may also be used to perform exportation and importation without having an intermediary file.

Other embodiments may use forms other than XML as an intermediary form. For example, ASCII or SGML may be used to convey information in an organized way. In addition, the use of XML may foster the use of a specialized language based on XML that may also be used. Though a preferred embodiment layered import and export logic over the model by deriving classes from the model classes and adding import and export methods thereto, other approaches may be used. Among other things, the derived class may have both import and export methods for a given object type. In addition, the logic

may be layered in other ways so that they exploit the gets and sets of the model data yet provide a consistent interface to the tools so that the import and export utility does not need to change when the model changes.

The exemplary embodiment above used Active X to convey client requests as COM + arguments, but other approaches may be substituted. For example, the browser may interact with the ASP page to invoke the applications as needed. The imports and export requests may then be performed through corresponding HTML pages.

In general, it should be emphasized that the various components of embodiments of the present invention can be implemented in hardware, software or a combination thereof. In such embodiments, the various components and steps would be implemented in hardware and/or software to perform the functions of the present invention. Any presently available or future developed computer software language and/or hardware components can be employed in such embodiments of the present invention. For example, at least some of the functionality mentioned above could be implemented using the C, C++, or any assembly language appropriate in view of the processor(s) being used. It could also be written in an interpretive environment such as Java and transported to multiple destinations to various users.

It is also to be appreciated and understood that the specific embodiments of the invention described hereinbefore are merely illustrative of the general principles of the invention. Various modifications may be made by those skilled in the art consistent with the principles set forth hereinbefore.

Having described an exemplary embodiment, it should be apparent to persons of ordinary skill in the art that changes may be made to the embodiment described without departing from the spirit and scope of the invention.

Appendix A

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE ENTITYDETAILS SYSTEM "entity.dtd" [
<!--
    Description :
    Author :
-->
]>
```

```
<ENTITYDETAILS>
   <ENTITY>
      <ENT_NAMESPACE>FAB300</ENT_NAMESPACE>
      <ENT_NAME>CLN201</ENT_NAME>
      <ENT_DATAGROUP/>
      <ENT_TPL_NAMESPACE>FAB300</ENT_TPL_NAMESPACE>
      <ENT_TPL_NAME>san_MasterEntityTemplate</ENT_TPL_NAME>
      <ENT_TPL_VERSION>1</ENT_TPL_VERSION>
      <OBJECT_STATUS>1</OBJECT_STATUS>
      <ENT_TYPE>Equipment</ENT_TYPE>
      <DESCRIPTION/>
      <ENT_STATE>ENGINEERING</ENT_STATE>
      <PM_FLAG>0</PM_FLAG>
      <WORKFLOW_NS>FAB300</WORKFLOW_NS>
      <WORKFLOW_NM>An user flow</WORKFLOW_NM>
      <WORKFLOW_VER>4</WORKFLOW_VER>
      <BEHAVIOR_MODEL_NS>FAB300</BEHAVIOR_MODEL_NS>
      <BEHAVIOR_MODEL>san_test_MB</BEHAVIOR_MODEL>
      <BEHAVIOR_MODEL_VER>1</BEHAVIOR_MODEL_VER>
      <PARENT_NS/>
      <PARENT_ENT/>
      <COMPONENT_ROLE/>
      <CHECKLIST_NS/>
      <CHECKLIST_NAME/>
      <CHECKLIST_VER>0</CHECKLIST_VER>
      <ATTRIBUTES>
         <ATTRIBUTETYPE>3</ATTRIBUTETYPE>
         <ATTRIBUTENAME>Equipment ID</ATTRIBUTENAME>
         <ATTRIBUTEINDEX>0</ATTRIBUTEINDEX>
         <DESCRIPTION/>
         <KEEPHISTFLAG>1</KEEPHISTFLAG>
         <PUBLISHFLAG>0</PUBLISHFLAG>
         <RESTRICTRULE>0</RESTRICTRULE>
         <MANDATORYFLAG>0</MANDATORYFLAG>
         <NAMEDTYPEFLAG>0</NAMEDTYPEFLAG>
         <NAMEDTYPENS/>
         <NAMEDTYPENAME/>
         <ATTRIBUTEVALUE/>
         <CDOSTRING>03</CDOSTRING>
      </ATTRIBUTES>
      <CATEGORIES>
         <CLASSIFTYPE>CSIM_ENTITY</CLASSIFTYPE>
         <NAMESPACE>FAB300</NAMESPACE>
         <NAME>Etech/Prc</NAME>
      </CATEGORIES>
   </ENTITY>
   <ENTITY>
      <ENT_NAMESPACE>FAB300</ENT_NAMESPACE>
      <ENT_NAME>CLN202</ENT_NAME>
      <ENT_DATAGROUP/>
      <ENT_TPL_NAMESPACE>FAB300</ENT_TPL_NAMESPACE>
      <ENT_TPL_NAME>san_MasterEntityTemplate</ENT_TPL_NAME>
      <ENT_TPL_VERSION>1</ENT_TPL_VERSION>
      <OBJECT_STATUS>1</OBJECT_STATUS>
      <ENT_TYPE>Equipment</ENT_TYPE>
      <DESCRIPTION/>
      <ENT_STATE>DOWN NON SCHED</ENT_STATE>
      <PM_FLAG>0</PM_FLAG>
      <WORKFLOW_NS/>
```

```
        <WORKFLOW_NM/>
        <WORKFLOW_VER>0</WORKFLOW_VER>
        <BEHAVIOR_MODEL_NS>FAB300</BEHAVIOR_MODEL_NS>
        <BEHAVIOR_MODEL>san_test_MB</BEHAVIOR_MODEL>
        <BEHAVIOR_MODEL_VER>1</BEHAVIOR_MODEL_VER>
        <PARENT_NS/>
        <PARENT_ENT/>
        <COMPONENT_ROLE/>
        <CHECKLIST_NS/>
        <CHECKLIST_NAME/>
        <CHECKLIST_VER>0</CHECKLIST_VER>
        <ATTRIBUTES>
            <ATTRIBUTETYPE>3</ATTRIBUTETYPE>
            <ATTRIBUTENAME>Equipment ID</ATTRIBUTENAME>
            <ATTRIBUTEINDEX>0</ATTRIBUTEINDEX>
            <DESCRIPTION/>
            <KEEPHISTFLAG>1</KEEPHISTFLAG>
            <PUBLISHFLAG>0</PUBLISHFLAG>
            <RESTRICTRULE>0</RESTRICTRULE>
            <MANDATORYFLAG>0</MANDATORYFLAG>
            <NAMEDTYPEFLAG>0</NAMEDTYPEFLAG>
            <NAMEDTYPENS/>
            <NAMEDTYPENAME/>
            <ATTRIBUTEVALUE/>
            <CDOSTRING>03</CDOSTRING>
        </ATTRIBUTES>
        <CATEGORIES>
            <CLASSIFTYPE>CSIM_ENTITY</CLASSIFTYPE>
            <NAMESPACE>FAB300</NAMESPACE>
            <NAME>Etech/Prc</NAME>
        </CATEGORIES>
    </ENTITY>
</ENTITYDETAILS>
```

Appendix B

ENTITY.DTD - Document Type Definition for Entity.

```
<!ELEMENT ENTITYDETAILS (ENTITY)* >
<!ELEMENT ENTITY
(ENT_NAMESPACE|ENT_NAME|ENT_DATAGROUP|ENT_TPL_NAMESPACE|ENT_TPL_NAME|EN
T_TPL_VERSION|OBJECT_STATUS|ENT_TYPE|DESCRIPTION|ENT_STATE|PM_FLAG|WORK
FLOW_NS|WORKFLOW_NM|WORKFLOW_VER|BEHAVIOR_MODEL_NS|BEHAVIOR_MODEL|BEHAV
IOR_MODEL_VER|PARENT_NS|PARENT_ENT|COMPONENT_ROLE|CHECKLIST_NS|CHECKLIS
T_NAME|CHECKLIST_VER|ATTRIBUTES|CATEGORIES)* >
<!ELEMENT ENT_NAMESPACE (#PCDATA)* >
<!ELEMENT ENT_NAME (#PCDATA)* >
<!ELEMENT ENT_DATAGROUP (#PCDATA)* >
<!ELEMENT ENT_TPL_NAMESPACE (#PCDATA)* >
<!ELEMENT ENT_TPL_NAME (#PCDATA)* >
<!ELEMENT ENT_TPL_VERSION (#PCDATA)* >
<!ELEMENT OBJECT_STATUS (#PCDATA)* >
<!ELEMENT ENT_TYPE (#PCDATA)* >
<!ELEMENT DESCRIPTION (#PCDATA)* >
<!ELEMENT ENT_STATE (#PCDATA)* >
<!ELEMENT PM_FLAG (#PCDATA)* >
<!ELEMENT WORKFLOW_NS (#PCDATA)* >
<!ELEMENT WORKFLOW_NM (#PCDATA)* >
<!ELEMENT WORKFLOW_VER (#PCDATA)* >
<!ELEMENT BEHAVIOR_MODEL_NS (#PCDATA)* >
<!ELEMENT BEHAVIOR_MODEL (#PCDATA)* >
<!ELEMENT BEHAVIOR_MODEL_VER (#PCDATA)* >
<!ELEMENT PARENT_NS (#PCDATA)* >
<!ELEMENT PARENT_ENT (#PCDATA)* >
<!ELEMENT COMPONENT_ROLE (#PCDATA)* >
<!ELEMENT CHECKLIST_NS (#PCDATA)* >
<!ELEMENT CHECKLIST_NAME (#PCDATA)* >
<!ELEMENT CHECKLIST_VER (#PCDATA)* >
<!ELEMENT ATTRIBUTES
(ATTRIBUTETYPE|ATTRIBUTENAME|ATTRIBUTEINDEX|DESCRIPTION|KEEPHISTFLAG|PU
BLISHFLAG|RESTRICTRULE|MANDATORYFLAG|NAMEDTYPEFLAG|NAMEDTYPENS|NAMEDTYP
ENAME|ATTRIBUTEVALUE|CDOSTRING)* >
<!ELEMENT ATTRIBUTETYPE (#PCDATA)* >
<!ELEMENT ATTRIBUTENAME (#PCDATA)* >
<!ELEMENT ATTRIBUTEINDEX (#PCDATA)* >
<!ELEMENT KEEPHISTFLAG (#PCDATA)* >
<!ELEMENT PUBLISHFLAG (#PCDATA)* >
<!ELEMENT RESTRICTRULE (#PCDATA)* >
<!ELEMENT MANDATORYFLAG (#PCDATA)* >
<!ELEMENT NAMEDTYPEFLAG (#PCDATA)* >
<!ELEMENT NAMEDTYPENS (#PCDATA)* >
<!ELEMENT NAMEDTYPENAME (#PCDATA)* >
<!ELEMENT ATTRIBUTEVALUE (#PCDATA)* >
<!ELEMENT CDOSTRING (#PCDATA)* >
<!ELEMENT CATEGORIES (CLASSIFTYPE|NAMESPACE|NAME)* >
<!ELEMENT CLASSIFTYPE (#PCDATA)* >
<!ELEMENT NAMESPACE (#PCDATA)* >
<!ELEMENT NAME (#PCDATA)* >
```

What is claimed is:

1. An MES application on a computer readable medium, the MES application having at least one model object, the data therefor persistently stored on a database, and wherein the one model object includes logic to receive an XML expression of model object data and to load the XML-expression of the data in the database.

2. The MES application of claim 1 wherein the logic to receive an XML expression includes logic to receive an importation policy for specifying whether or not database data should be over-ridden with the XML-expression data and logic to load the XML-expression if the policy states that database data should be over-ridden.

3. The MES application of claim 1 wherein the logic to receive and to load includes logic to parse the XML expression and to convert XML tag labels to database address information, and to convert XML element contents to database values.

4. The MES application of claim 1 wherein the at least one model object is constructed from a class that is derived from an MES model object class, and wherein the class so derived inherits all functionality from the MES model object class and wherein the class further includes import method logic to receive an XML expression of model object data and to load the XML-expression of the data in the database.

5. An MES application on a computer readable medium, the MES application having at least one model object, the data therefor persistently stored on a database, and wherein the one model object includes logic to create an XML expression of model object data in the database.

6. The MES application of claim 5 wherein the logic to create includes logic to create XML tag information from database reference information for the database entity holding the model object data, and logic to retrieve the model object data from the database and to insert the retrieved data as XML element contents.

7. The MES application of claim 1 wherein the at least one model object is constructed from a class that is derived from an MES model object class, and wherein the class so derived inherits all functionality from the MES model object class and wherein the class further includes export method logic to create an XML expression of model object data in the database.

8. A method of importing data into an MES application having at least one model object, comprising:

a. forming an XML expression of the object data;

b. sending the XML expression to the MES application;

c. the MES application parsing the XML expression to determine the object type, the object parts, and the object data values;

d. the MES application loading the determined object type, the object parts, and the object data values into a database corresponding to the MES application.

9. The method of claim 8 further comprising
forming a DTD expression of the object class definition;
validating the XML expression against the DTD expression.

10. A method of exporting object data in which a model object has data stored in a database, comprising:

a. forming XML tag labels from database address information for the object data to be imported;

b. accessing the database to obtain the values of the stored object data;

c. using the retrieved values as contents for the XML element corresponding to the stored object data;

d. providing an XML document created from (a)-(c) as an exported version of the model object data.

11. The method of claim 6 further comprising creating a DTD expression of the object class definition for the object to be exported.

12. An MES application on a computer readable medium, the MES application having at least one model object, the data therefor persistently stored on a database, and wherein the application has import logic layered over the model object, wherein the import logic defines an external interface for loading model object data, and wherein the import logic includes logic to invoke the model object to load the model object data into the database.

13. An MES application on a computer readable medium, the MES application having at least one model object, the data therefor persistently stored on a database, and wherein the application has export logic layered over the model object, wherein the export logic defines an external interface for obtaining model object data, and wherein the export logic includes logic to invoke the model object to obtain the model object data from the database.

14. A manufacturing execution system (MES), comprising:

a client computer having a user interface providing a control for exporting or importing selected model objects of the MES and for constructing a client request to request export or import of selected objects; and

a server computer having the MES application executing thereon, the MES application organized as a relationship of model objects, with each model object having data therefor stored in a database, wherein the application includes import and export logic, responsive to the client request, for forming an XML expression of model object data in response to an export request, and for loading data expressed in XML form into the database in response to an import request.

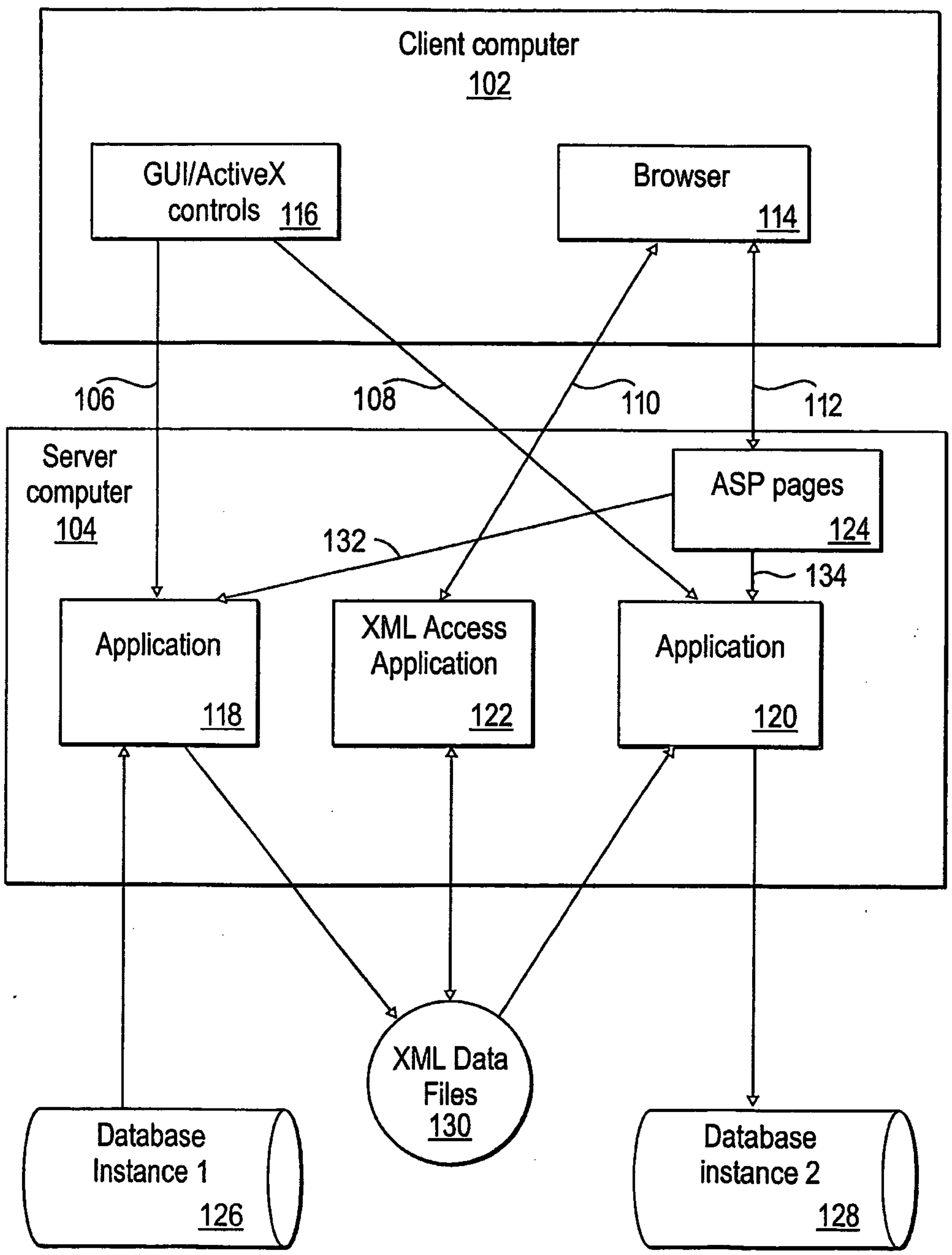

Client computer
102
GUI/ActiveX controls 116
Browser
114
106
108
110
112
Server computer
104
ASP pages
124
132
134
Application
118
XML Access Application
122
Application
120
XML Data Files
130
Database Instance 1
126
Database instance 2
128

Fig. 1

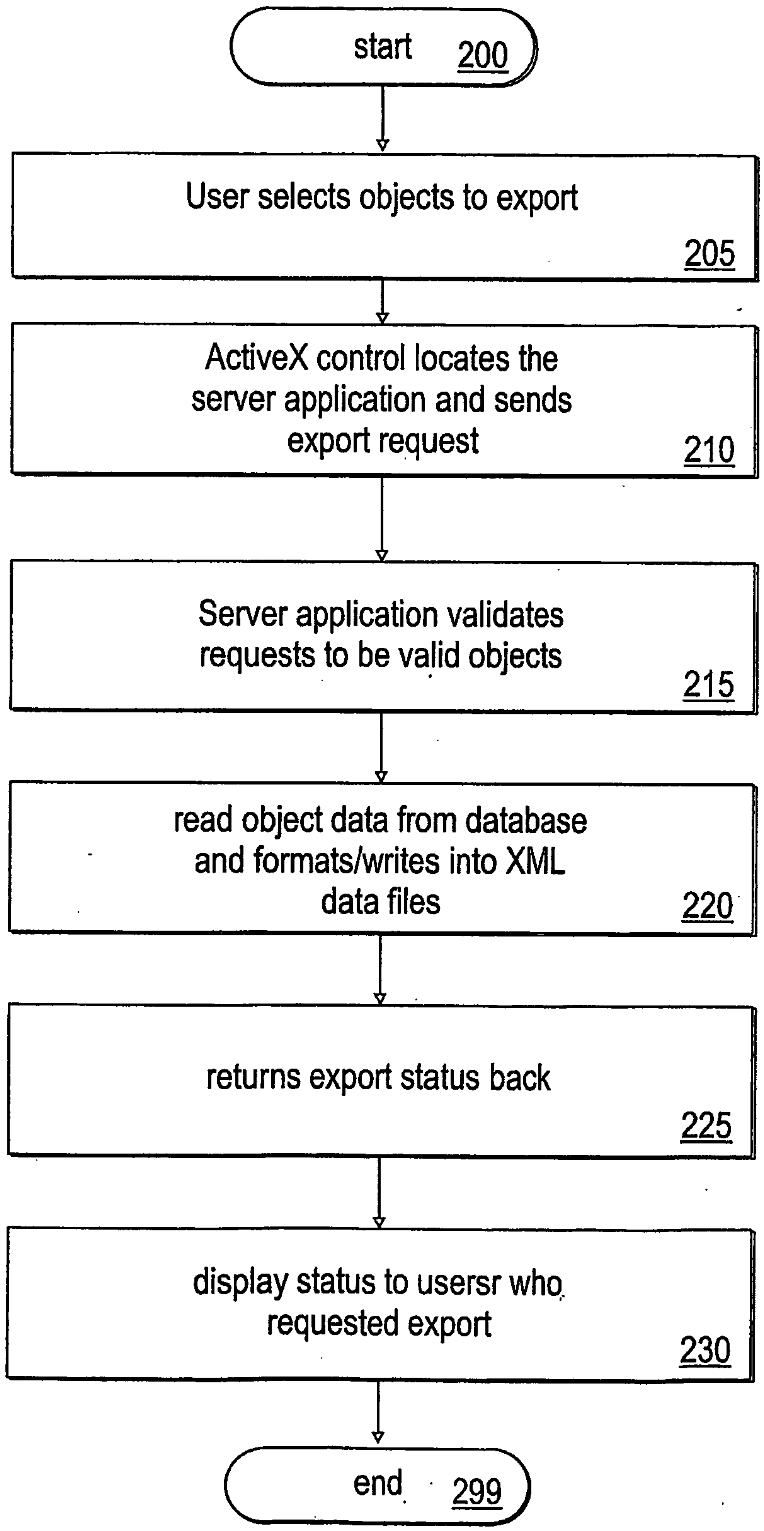

start 200
User selects objects to export 205
ActiveX control locates the server application and sends export request 210
Server application validates requests to be valid objects 215
read object data from database and formats/writes into XML data files 220
returns export status back 225
display status to usersr who requested export 230
end 299

Fig. 2

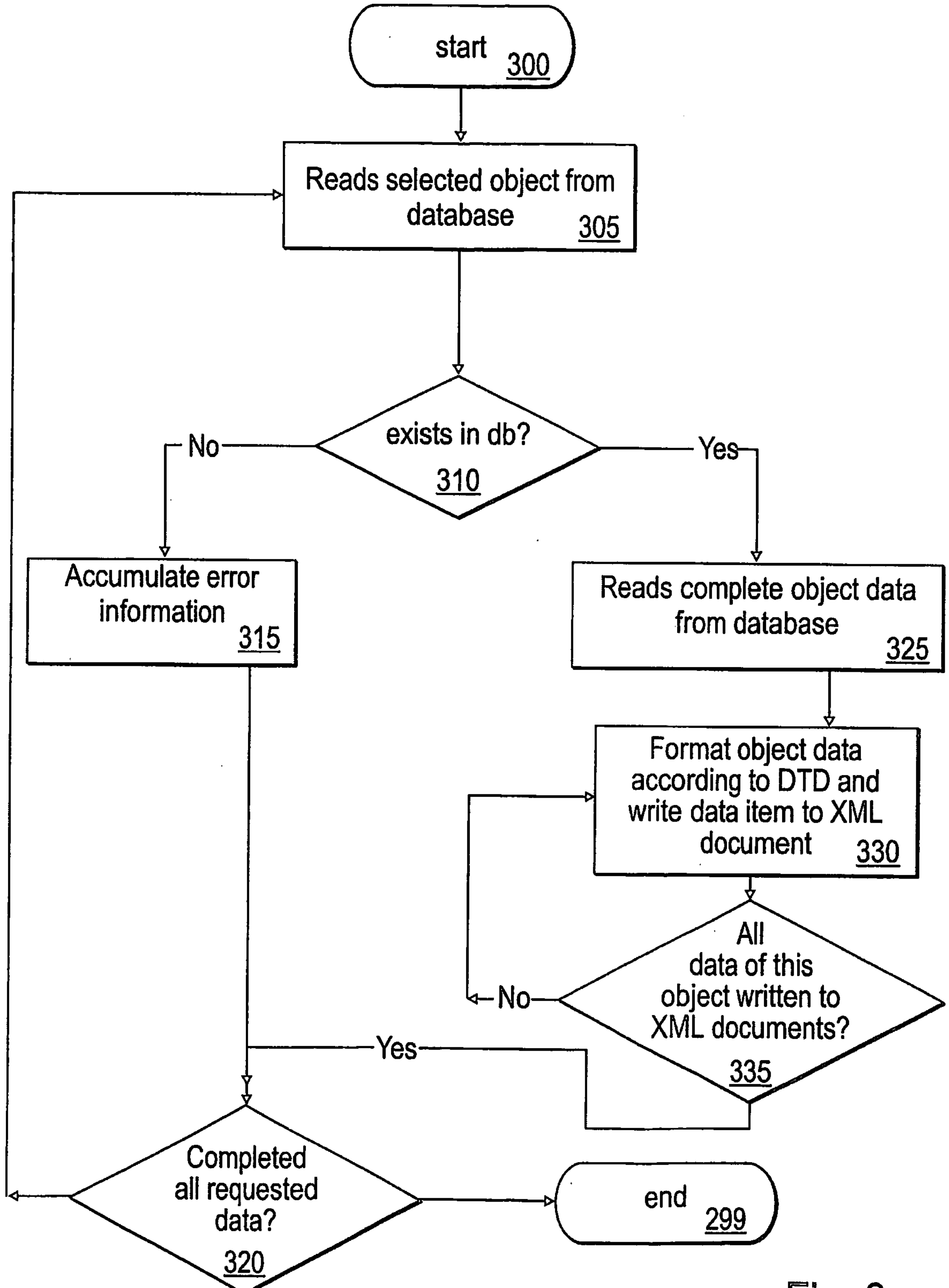

start 300
Reads selected object from database 305
exists in db? 310
No
Yes
Accumulate error information 315
Reads complete object data from database 325
Format object data according to DTD and write data item to XML document 330
All data of this object written to XML documents? 335
No
Yes
Completed all requested data? 320
end 299

Fig. 3

4/8

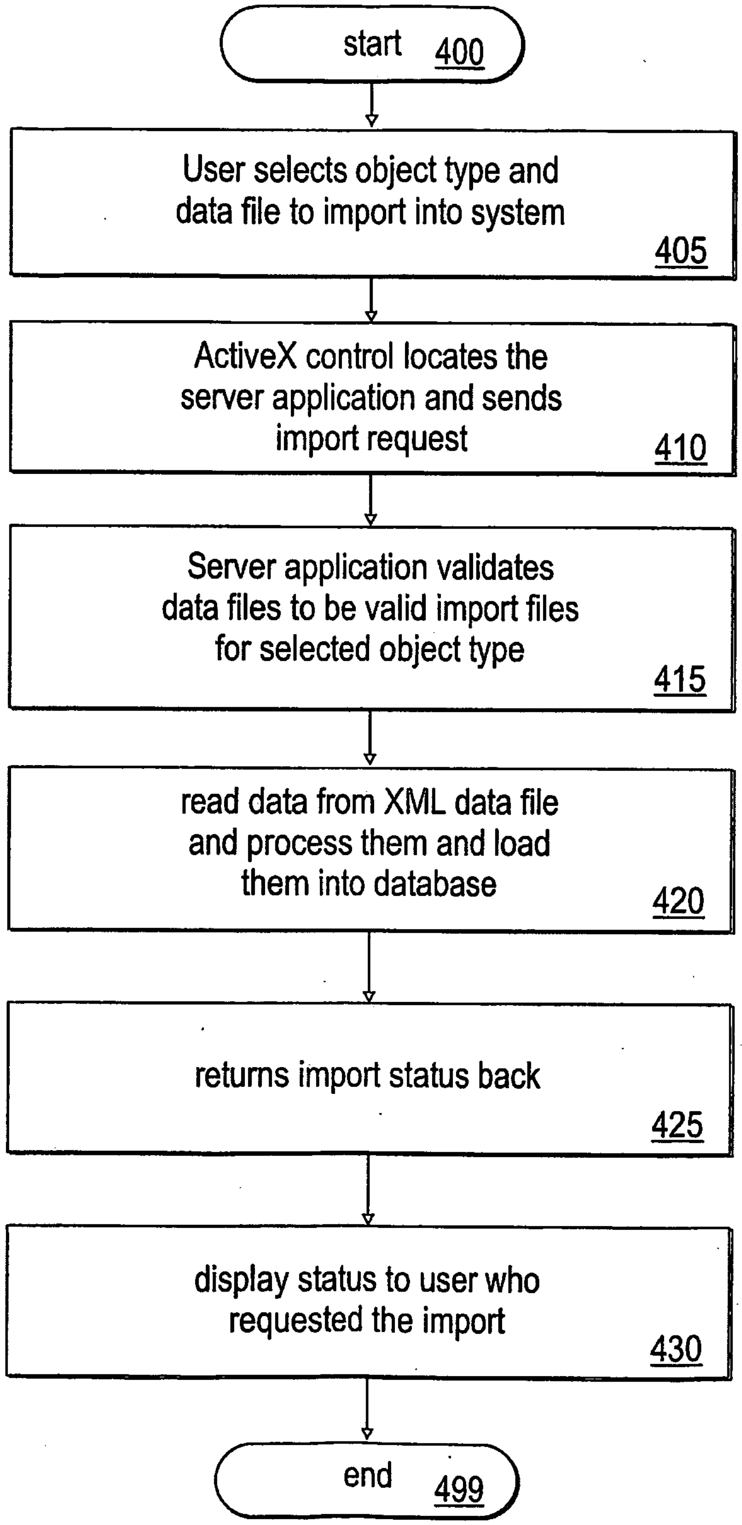

start 400
User selects object type and data file to import into system 405
ActiveX control locates the server application and sends import request 410
Server application validates data files to be valid import files for selected object type 415
read data from XML data file and process them and load them into database 420
returns import status back 425
display status to user who requested the import 430
end 499

Fig. 4

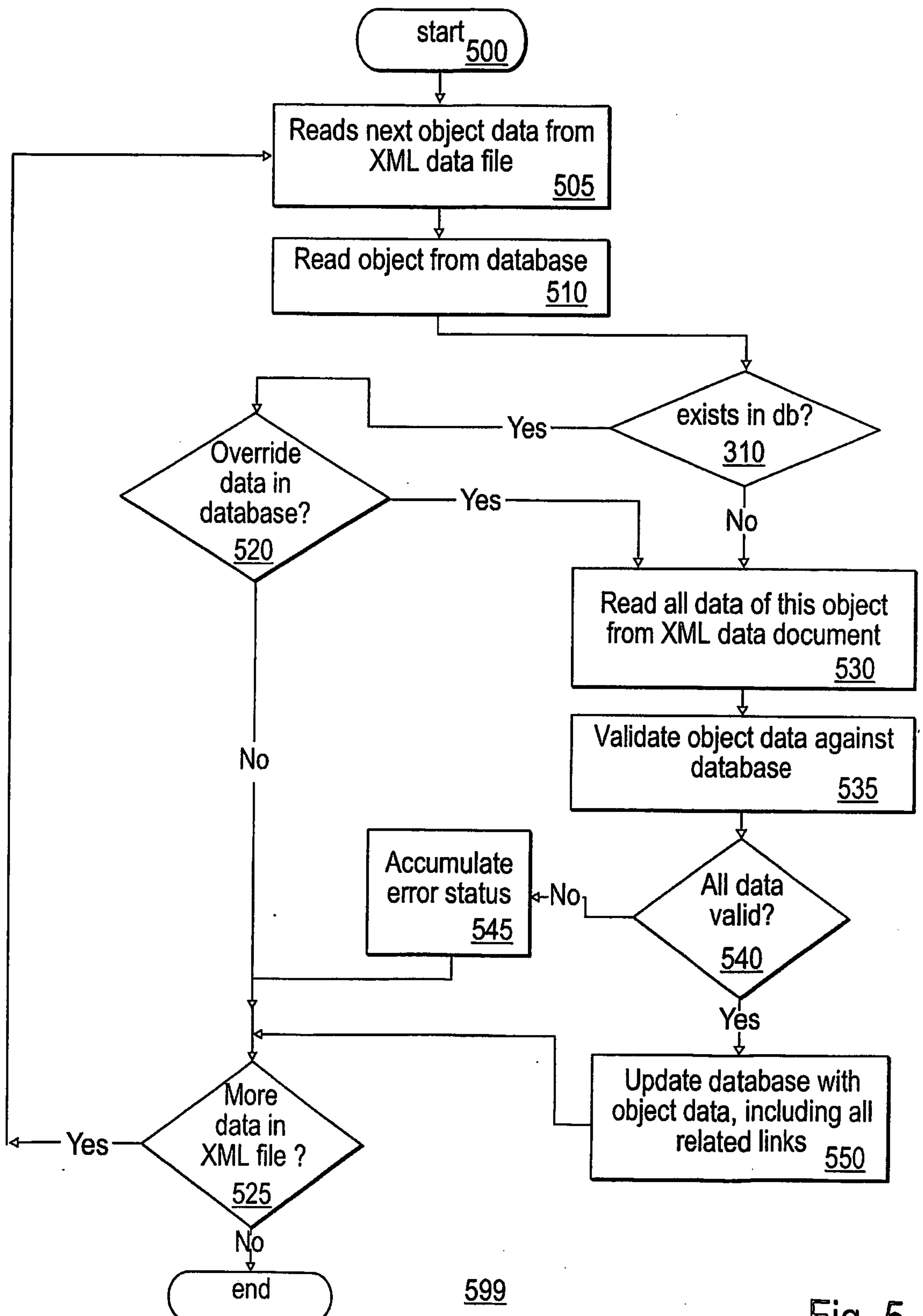

start 500
Reads next object data from XML data file 505
Read object from database 510
exists in db? 310
Yes
No
Override data in database? 520
Yes
No
Read all data of this object from XML data document 530
Validate object data against database 535
All data valid? 540
No
Yes
Accumulate error status 545
Update database with object data, including all related links 550
More data in XML file ? 525
Yes
No
end
599

Fig. 5

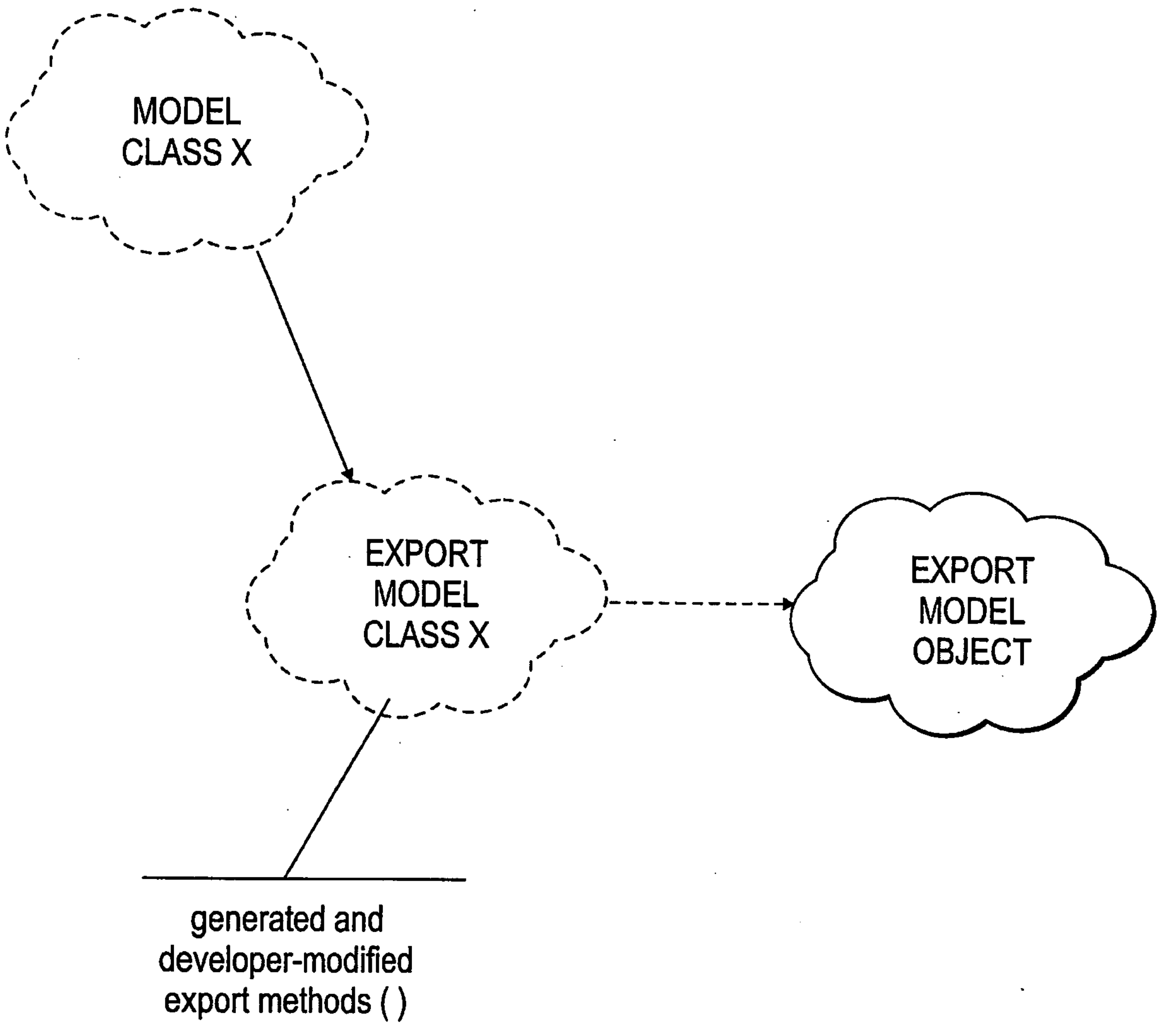

MODEL
CLASS X
EXPORT
MODEL
CLASS X
EXPORT
MODEL
OBJECT
generated and
developer-modified
export methods ( )

Fig. 6A

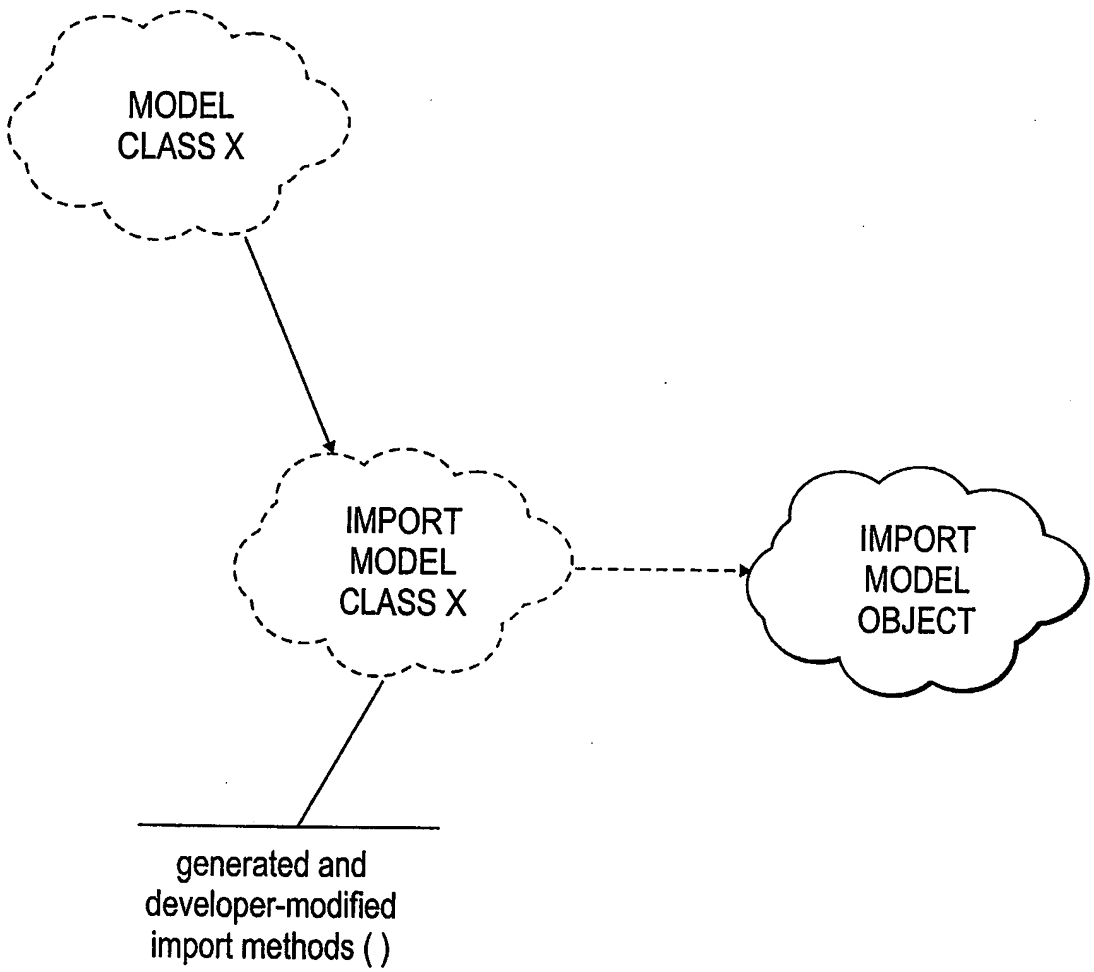

MODEL
CLASS X
IMPORT
MODEL
CLASS X
IMPORT
MODEL
OBJECT
generated and
developer-modified
import methods ( )

Fig. 6B

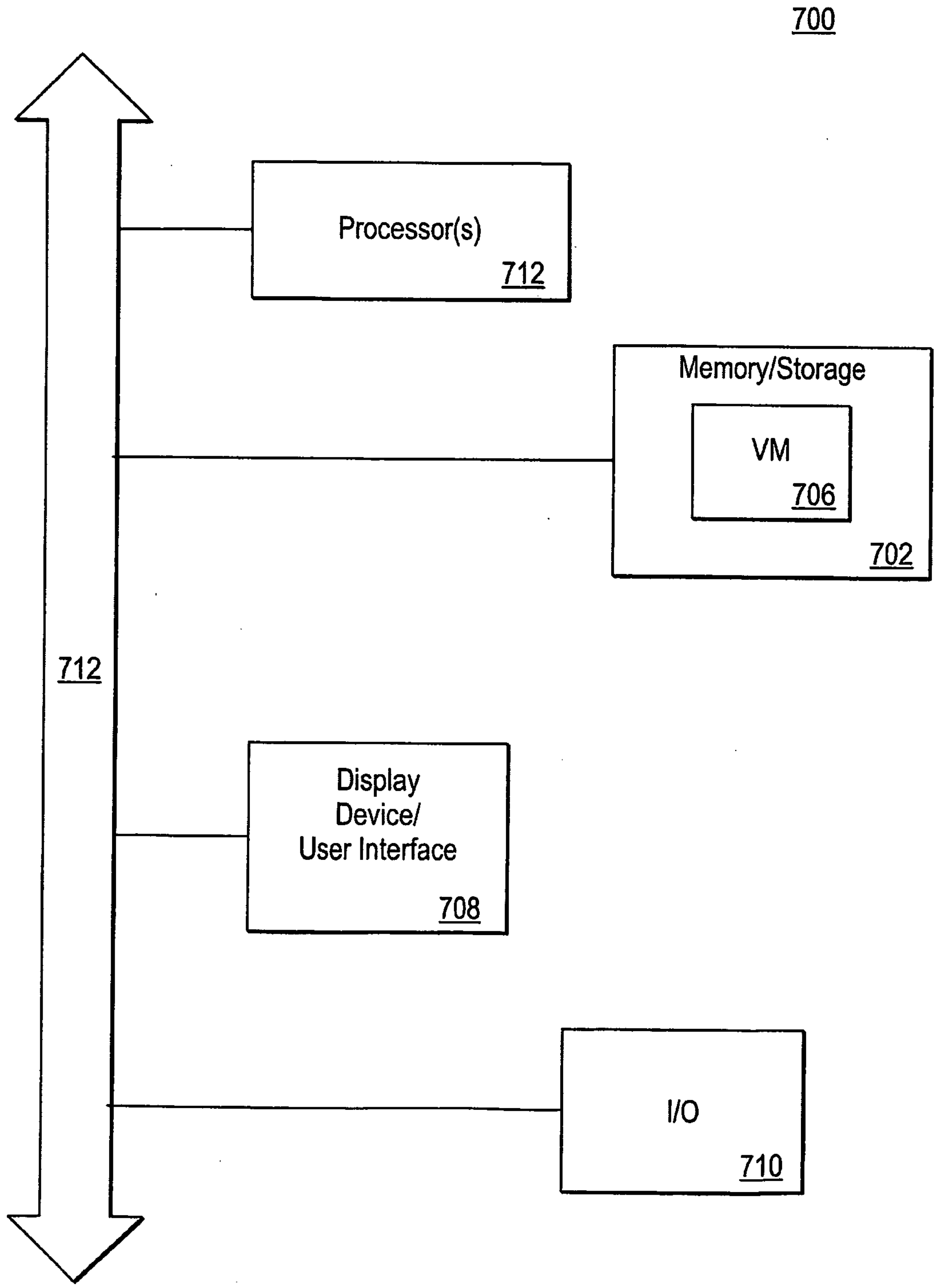

700
Processor(s)
712
Memory/Storage
VM
706
702
712
Display
Device/
User Interface
708
I/O
710

Fig. 7

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US 01/22833

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G05B19/418

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G05B

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
E	WO 01 57823 A (DOMAIN LOGIX CORP) 9 August 2001 (2001-08-09) page 3, line 10 -page 15, line 10; figures 3-5	1-14
E	WO 02 17150 A (PRI AUTOMATION INC) 28 February 2002 (2002-02-28) page 6, line 11 -page 14, line 12; figures 1-4	1-14
X	WO 01 52055 A (WIND RIVER SYSTEMS INC) 19 July 2001 (2001-07-19) page 5, line 2 -page 14, line 22; figures 1-9	1-14

-/--

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
26 September 2002	04/10/2002

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Tran-Tien, T

Form PCT/ISA/210 (second sheet) (July 1992)

International Application No
PCT/US 01/22833

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
X	US 6 253 366 B1 (MUTSCHLER III EUGENE OTTO) 26 June 2001 (2001-06-26) page 4, line 18 -page 16, line 29; figures 1-16 ---	1-14
A	WO 95 34866 A (HELM ANDREW RICHARD ;FORTIN DENNIS (CA); BERDYCH JULIAN (CA); BOUC) 21 December 1995 (1995-12-21) abstract ---	1-14
A	SUZUKI J ET AL: "Toward the interoperable software design models: quartet of UML, XML, DOM and CORBA" , PROCEEDINGS IEEE INTERNATIONAL SOFTWARE ENGINEERING STANDARDS SYMPOSIUM, XX, XX, PAGE(S) 163-172 XP002170829 abstract ---	1-14
A	KLEIN B: "APPLICATION DEVELOPMENT XML MAKES OBJECT MODELS MORE USEFUL" , INFORMATIONWEEK, MANHASSET, NY, US, PAGE(S) 1A-6A XP002935959 ISSN: 8750-6874 the whole document -----	1-14

INTERNATIONAL SEARCH REPORT
Information on patent family members

International Application No
PCT/US 01/22833

Patent document cited in search report		Publication date	Patent family member(s)		Publication date
WO 0157823	A	09-08-2001	AU	3316401 A	14-08-2001
			WO	0157823 A2	09-08-2001
			US	2002026514 A1	28-02-2002
WO 0217150	A	28-02-2002	AU	8548001 A	04-03-2002
			WO	0217150 A1	28-02-2002
			US	2002095644 A1	18-07-2002
WO 0152055	A	19-07-2001	AU	2759101 A	24-07-2001
			WO	0152055 A2	19-07-2001
US 6253366	B1	26-06-2001	NONE		
WO 9534866	A	21-12-1995	AU	2759595 A	05-01-1996
			WO	9534866 A1	21-12-1995